26 | 平台产品研发：三个月完成千人规模的产品要怎么做？

2019-12-17 石雪峰

讲述：石雪峰
时长 20:34 大小 18.85M

你好，我是石雪峰。

虽然我们之前聊了这么多的平台建设思路，但是，可能很多人都没有机会经历一个平台从构思到开发、再到推广落地的完整过程。

如果要开发一个千人使用的 DevOps 产品，需要多长时间呢？你可能会说需要半年甚至是更长的时间，我之前也是这么觉得的。

但是，2018 年，在启动流水线平台建设的时候，老板“大手一挥”，要求在三个月内见到成效，我都快惊呆了。

因为，我们要真正地从零开始：原型图都没有一张，代码都没有一行，临时组建的一个草台班子还分散在北京、上海两地，团队成员之前都没怎么打过招呼，这能行吗？

今天，我想给你分享的就是这个真实的故事。我来跟你一起复盘下这次“急行军”的历程，看看我们做对了什么，又做错了什么，有哪些干货是可以拿来就用的，又有哪些“坑”是你一定要努力回避的。

其实，作为一个非专业的 DevOps 产品经理，你终将面对这样的挑战，但你要相信，**只要开始去做了，就没有什么是不可能的**。

项目启动

时间回到一年前，当时我所在的这个“草台班子”是个啥情况呢？团队组成是这样的：两个后台开发在北京，一个半前端开发在上海，还有一个基础设施工程师和一个流水线开发工程师，再加上半个全能打杂的产品经理，也就是我，满打满算一共 6 个人。

项目从 11 月中旬开始构思，12 月初开启动会，当时，除了我之外，没有任何人清楚我们要做的到底是个什么玩意儿。这该怎么办呢？

玩过游戏的同学应该都知道打好开局有多重要，所以，为了这个 Kickoff 会议，我事先做了大量的准备工作，其中就包括 0.2 版本的产品原型图。与其说是一个原型图，不如说就是一个草稿，简陋得不能再简陋了。

项目的 Kickoff 会议是项目组成员和未来产品的第一次见面，留下一个积极的印象非常重要。所以，从第一刻开始，我就铆足了精神。

首先，我发出了一封热情洋溢的会议邀请。在会议邀请中，我仔细地陈述了我们为什么要做这件事，为什么是现在，为什么不做不行。

在正式开会的时候，我再一次明确了项目的重要性和紧急性，并给大家演示了第一版的系统原型图（没错，就是简陋到极致的刚刚的这张原型图）。

即便这样，三个月的工期也让大家非常焦虑。为了缓解紧张情绪，证明这个项目的可行性，我还做了两件事：

1. 搭建了一个系统 demo，几个简单的页面；
2. 由于用到了另外一个开源产品的核心技术，于是，我就对这个技术进行了简单演示。

虽然我自己心里对这个计划也相当“打鼓”，但我还是希望告诉大家，这并不是不可能的任务，努力帮助大家树立信心。

在项目启动会上，团队达成了两个非常关键的结论：**一个是系统方案选型；另一个是建立协作机制**。

首先，由于时间紧任务重，我们决定使用更易于协作的前后端分离的开发模式。后来，事实证明，这是一个非常明智的选择。这不仅大幅提升了开发效率，也大大降低了之后向移动端迁移的成本。在开发移动端产品的时候，后端接口大部分都可以直接拿来使用。

在技术框架方面，由于大家对前后端分离的模式达成了共识，我们就采用 **Python+Django+VUE** 的方式来做。你可能会问，为啥不用基于 Java 的 Spring 系列呢？因为我觉得，对于内部系统来说，这些典型的框架应付起来基本都绰绰有余，关键还是要选你熟悉的、易上手的那个。从这个角度来看，Python 显然有着得天独厚的优势。即便之前只是写写脚本，想要上手 Python 也不是一件困难的事情。

在项目协作方面，我等会儿会专门提到，由于团队成员分散在北京、上海两地，彼此之间不够熟悉和信任，所以，**建立固定的沟通机制就非常重要**。

至少，在项目初期，我们每周都要开两次电话会议：

一次是面向全员的。一方面同步项目的最新进展，另一方面，也给大家一些紧迫感，让大家觉得“其他人都在按照计划执行，自己也不能落后”。

另外一次是面向跨地域骨干的。这主要还是为了增进联系，并且对一些核心问题进行二次的进展确认。不拉上全员，也是为了避免过多地浪费项目成员的时间。

最后，项目毕竟还是有一些技术风险的，所以还需要启动预研。我们这个项目的主要风险是在**前端交互**上。

这是一个从来没人实现过的场景，有大量的用户界面编排操作在里面。所以，我们专门指定了一位同学，让他啥也别想，一门心思地进行技术攻关。

事实证明，但凡能打硬仗的同事，在后来都是非常靠谱且独当一面的，这与年龄无关，哪怕是应届生，也同样如此。

讲到这里，我要先给你总结一下在项目启动阶段要重点关注的几件事情：

明确项目目标，树立团队的信心；

沟通开发模式和技术架构选型，以快速开发和简单上手为导向；

建立沟通渠道，保持高频联系；

识别项目的技术风险，提前开启专项预研。

开发策略

人类社会活动的每一个环节，都需要越来越多的人为了同一个目标推进工作，软件开发也不例外。那么，我们是怎么做的呢？

首先，就是**研发环境容器化**。

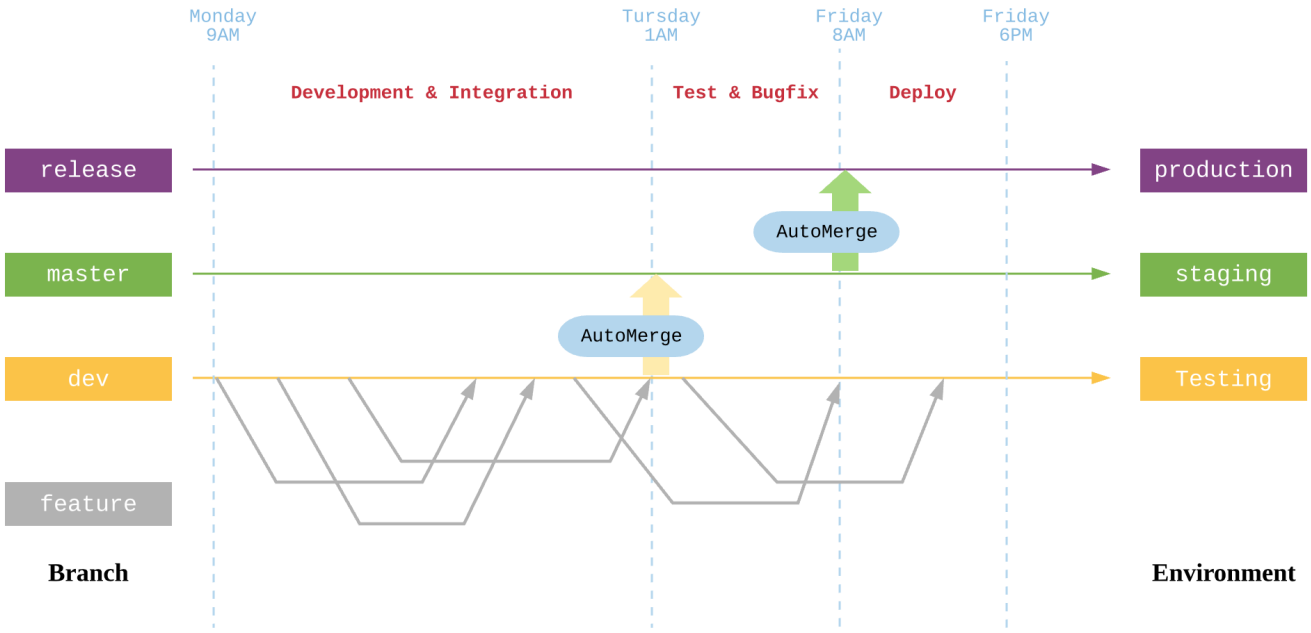
对于接触一个全新技术栈的开发来说，本地搭建一套完整可运行的环境总是绕不过去的坎。即便是对照着文档一步步操作，也总会有遗漏的地方。除此之外，项目依赖的各种中间件，哪怕稍微有一个版本不一致，最后一旦出现问题，就要查很久。

既然如此，为什么不一上来就采用标准化的环境呢？这就可以发挥容器技术的优势了。主力后台开发同学自己认领了这个任务，先在本地完成环境搭建并调试通过，接着把环境配置容器化。这样一来，新人加入项目后，几分钟就能完成一套可以工作的本地开发环境。即便后续要升级环境组件，比如 Django 框架版本，也非常简单，只要推送一个镜像上去，再重启本地环境就可以了。

其次，就是选择分支策略。虽然 DevOps 倡导的是主干开发，但是我们还是选择了“三分支”的策略，因为**我们搭建了三套环境**。

测试环境对应 dev 分支作为开发主线，所有新功能在特性分支开发，自测通过后，再通过 MR 到 dev 分支并部署到测试环境进行验收测试，一般验收测试由需求提出方负责。

接下来，定期每周两次从 dev 上 master 分支，master 分支对应了预发布环境，保证跟生产环境的一致性，数据也会定期进行同步。只有在预发布环境最终验收通过后，才具备上线生产环境的条件。通过将 master 分支合并到 release 分支，最后完成生产环境部署。这种分支策略的示意图如下：



为什么要采用三套环境的“三分支”策略呢？

这里的主要原因就是，团队处于组建初期，磨合不到位，经常会出现前后端配置不一致的情况。更何况，我们这个项目不只有前后端开发，还有核心原子业务开发，以及基础设施维护。任何一方的步调不一致，都会导致出现问题。

另外，内部平台开发往往有个通病，就是**没有专职测试**。这也能理解，总共才几个人的“草台班子”，哪来的测试资源啊？所以，基本上只能靠研发和产品把关。

但是，毕竟测试也是个专业的工种，这么一来，总会有各种各样的问题。再加上，产品需求本身就没有那么清晰、灵活多变，所以，**多一套环境，多一套安全**。

但不可否认的是，这种策略并非是最优解，只不过是适应当时场景下的可行方案。当团队磨合到位，而且也比较成熟之后，就可以简化一条分支和一套环境了。不过，前提是，**只有快速迭代，快速上线，才能发挥两套环境的优势**。

> Use what you build to build what you use.（使用你开发的工具来开发你的工具）

这是我们一以贯之的理念。既然是 DevOps 平台，那么团队也要有 DevOps 的样子，所以，作为一个全功能团队，研发自上线和研发自运维就发挥到了极致。

同时，我们并没有使用公司统一的上线流程，而是自己建立了一个标准化的上线流程并固化在工具里面，团队的每一个人都能完成上线动作。

这样一来，就不会再依赖于某个具体的人员了，这就保持了最大的灵活性。即便赶上大促封网，也不会阻塞正常的开发活动。

开发协作流程

仅仅是做到上面这几点，还不足以让整个团队高效运转起来，因为**缺少了最重要的研发协作流程**。

作为项目负责人，我花了很大的精力优化研发协作流程，制定研发协作规范。当这一切正常运转起来后，我发现，这些前期的投入都是非常值得的。

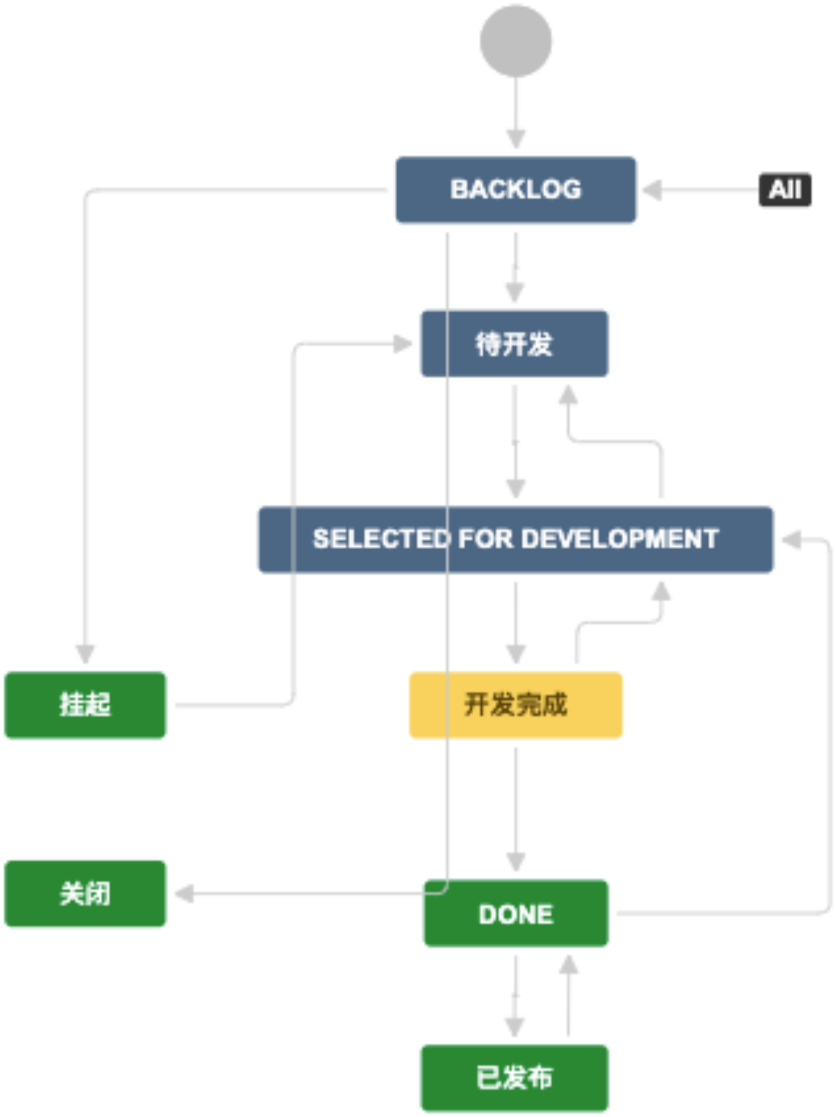
在工具层面，我们使用了 Jira。对于小团队来说，Jira 的功能就足够优秀了，可以满足大多数场景的需求。但是 Jira 的缺点在于，使用和配置门槛稍微有点高。因此，团队里面需要有一个熟悉 Jira 的成员，才能把这套方法“玩”下去。

在 Jira 里面，我们采用了**精益看板加上迭代**的方式，基本上两周一个迭代，保持开发交付的节奏。这种开发工作流刚好适配我们的分支策略和多环境部署。

需求统一纳入 Backlog 管理，当迭代开始时，就拖入待开发状态，研发挑选任务启动开发，并进入开发中。当开发完成后，也就意味着功能已经在测试环境部署。这个时候，就可以等待功能验收。只有在验收通过之后，才会发布到预发布环境。并经过二次验收后，最终上线发布给用户。

开发流程并不复杂，你可以看一下下面这两版流程图。

图片版：



文字版：

From	Transition		To
BACKLOG	准备开发 No Screen	→	待开发
	关闭 No Screen	→	关闭
	挂起 No Screen	→	挂起
SELECTED FOR DEVE...	已解决 No Screen	→	开发完成
	暂缓开发 No Screen	→	待开发
DONE	上线发布 No Screen	→	已发布
	驳回开发 No Screen	→	SELECTED FOR DEVE...
开发完成	验收通过 No Screen	→	DONE
	驳回开发 No Screen	→	SELECTED FOR DEVE...
待开发	启动开发 No Screen	→	SELECTED FOR DEVE...
挂起	ReOpen No Screen	→	待开发
已发布	撤销发布 No Screen	→	DONE
关闭	There are no transitions out of this status		

定义好开发工作流之后，接下来，就需要明确原则和规范了。对于一个新组建的团队来说，**规则是消除分歧和误解的最好手段，所以一定要让这些规则足够得清晰易懂**。比如，在我们内部就有一个“3-2-1”原则：

3：**创建**任务三要素

有详细的问题说明和描述

有清晰的验收标准

有具体的经办人和迭代排期

2：**处理**任务两要素

在开发中，代码变更要关联 Jira 任务号

在开发完成后，要添加 Jira 注释，说明改动内容和影响范围

1：**解决**任务一要素

问题报告人负责任务验收关闭

当然，团队规则远不止这几条。你要打造自己团队内部的规则，并且反复地强调规则，帮助大家养成习惯。这样一来，你会发现，**研发效率提升和自组织团队都会慢慢成为现实**。

除此之外，**你也不要高估人的主动性，期望每个人都能自觉地按照规则执行**。所以，定期和及时的提醒就非常必要。比如，每天增加定时邮件通知，告诉大家有哪些需求需要验收，有哪些可以上线发布，尽量让每个人都明白应该去哪里获取最新的信息。

另外，每次开周会时，都要强调规则的执行情况，甚至每天的站会也要按需沟通。只有保持短促、高频的沟通，才能产生理想的效果。

产品运营策略

关于产品运营策略，“酒香不怕巷子深”的理念已经有些过时了。想要一个产品获得成功，**团队不仅要做得好，还要善于运营和宣传，而这又是技术团队的一大软肋**。

开发团队大多只知道如何实现功能，却不知道应该怎么做产品运营。往往也正因为如此，团队很难获取用户的真实反馈，甚至开发了很多天才的功能，用户都不知道。产品开发变成了“自嗨”，这肯定不符合产品设计的初衷。

考虑到这些，我们在平台运营的时候，也采取了一些手段。我想提醒你的是，**很多事情其实没有没有多难，关键就看有没有想，有没有坚持做**。

比如，你可以建立内部用户沟通群，在产品初期尽量选择一些活跃的种子用户来试用。那些特别感兴趣、愿意尝试新事物、不断给你提建议的都是超级用户。这些用户未来都是各个团队中的“星星之火”，**在项目初期，你一定要识别出这些用户**。

另外，每一次上线都发布一个 release notes，并通过邮件和内部沟通群的方式通知全员，一方面可以宣传新功能，另一方面，也是很重要的一方面，**就是保持存在感的刷新**。你要让用户知道这个产品是在高速迭代的过程中的，而且每次都有不一样的新东西，总有一样会吸引到他们，或者让他们主动提出自己的问题。

在用户群里面，注意要及时响应用户的问题。你可以在团队内部建立 OnCall 机制，每周团队成员轮值解决一线用户的问题，既可以保证问题的及时收敛，也能让远离用户的开发真真切切地听到用户的声音。这样的话，在需求规划会和迭代回顾会的时候，开发就会更多地主动参与讨论。

以上这些都是比较常规的手段，在我们的产品运营中，还有两个方法特别有效，我也推荐给你。

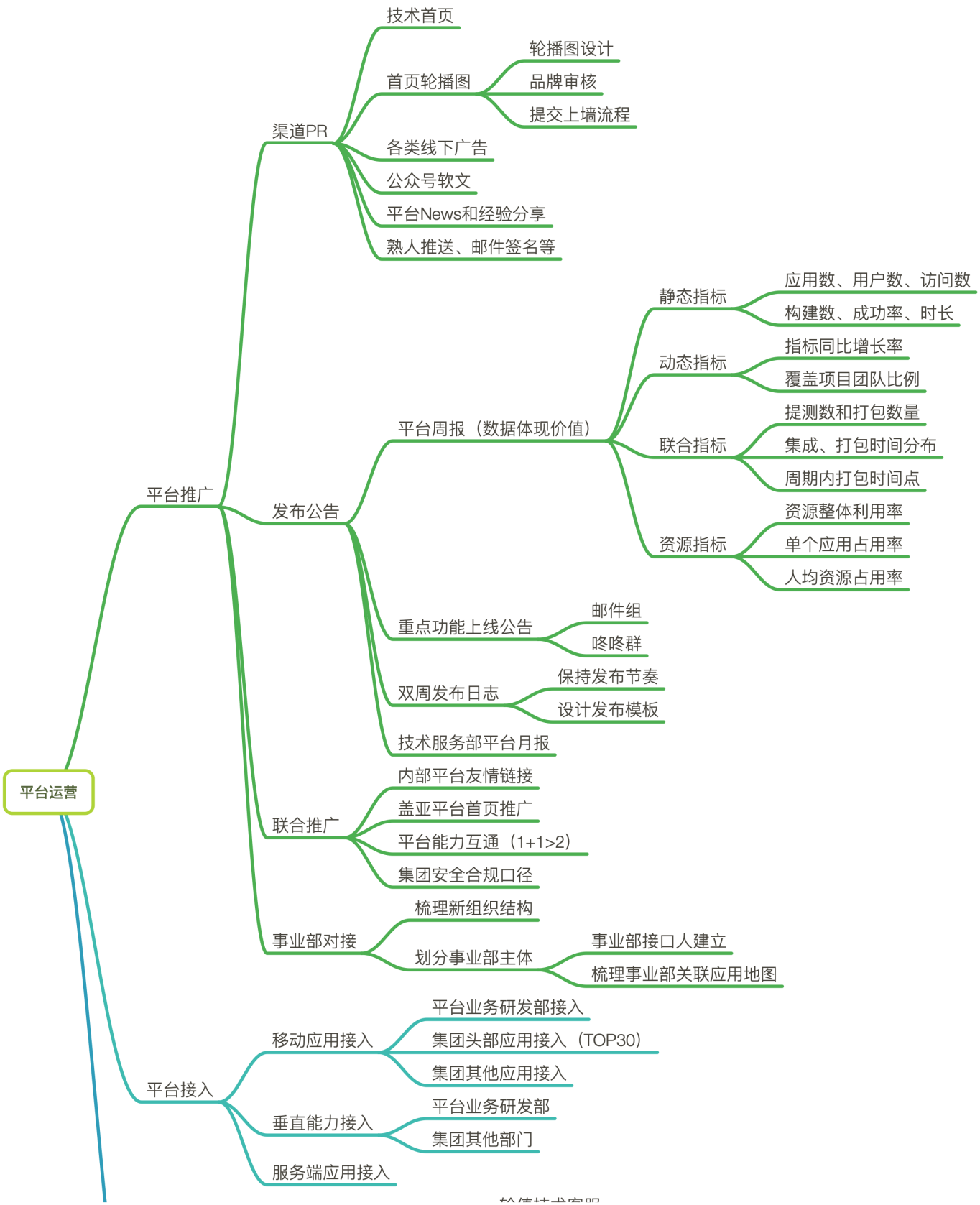
平台运营就跟打广告是一样的，越是在人流最大、关注度最高的地方打广告，效果也就越好。每个公司一般都有类似的首页，比如公司内部的技术首页、技术论坛、日常办公的 OA 系统等等，这些地方其实都会有宣传的渠道和入口。**你要做的就是找到这个入口，并联系上负责这个渠道的人员**。我们的产品就一度实现了热门站点的霸屏，宣传效果非常明显，用户量直线上升。

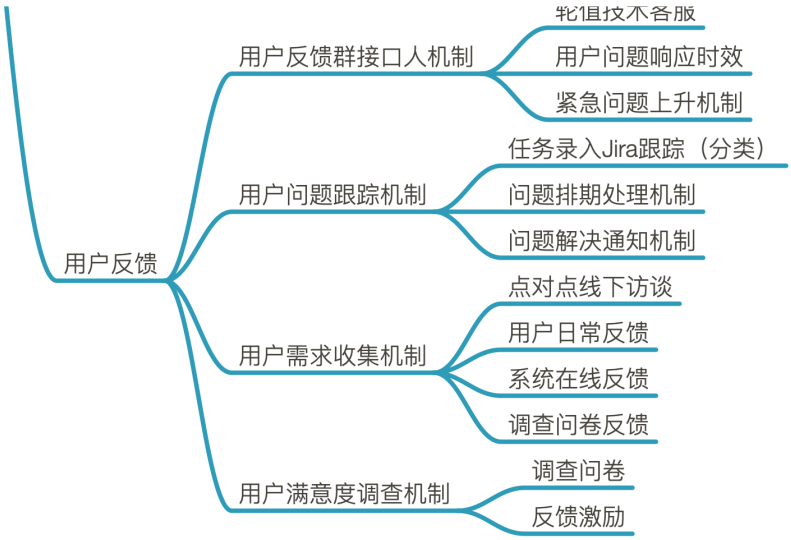
另一个方法有些取巧，但对于技术团队来说，也非常适用，那就是**通过技术分享的渠道来宣传产品**。

相信每个团队都会有定期的技术分享渠道，或者是技术公众号等，**你可以把平台的核心技术点和设计思想提炼出来，拟定一个分享话题，并在内部最大范围的技术分享渠道中进行分**

享。

很多时候，单纯地宣传一个产品，很多人是“不感冒”的。但是，如果你在讲一些新技术，并结合产品化落地的事情，对技术人员的吸引力就会大很多。所以，**换个思路做运营，也是提升产品知名度的好方法。**我把我之前总结的产品运营渠道和手段汇总成了一幅脑图，也分享给你。





团队文化建设

最后，我想再跟你简单聊聊团队文化建设的事情。毕竟，无论什么样的工具、流程、目标，最终都是依靠人来完成的。**如果忽略对人的关注，就等同于本末倒置，不是一个成熟的团队管理者应该做的事情**。我给你分享我的两点感受。

1. 让专业的人做专业的事情

很多时候，千万不要小看专业度这个事情。任何一个组织内部的职能都需要专业能力的支撑，这些专业能力都是量变引发的质变。

我举个最简单的例子，你还记得我在前面提到的 0.2 版本的原型草稿吗？实际上，到了 0.3 版本，引用前端工程师话来说，“原型做得比系统还漂亮”。这是为什么呢？难道是我这个“半吊子”产品经理突然开窍了吗？

显然不是。其实答案很简单，就是我去找了专业产品经理做外援，让他帮我改了两天的原型图。对于专业的人来说，这些事情再简单不过了。

找专业的人来做这些事情，不仅可以帮助你快速地跨越鸿沟，也能留下很多现成的经验，供你以后使用，这绝对不是一个人埋头苦干可以做得到的。

不仅是产品方面，技术领域也是一样的。**我们要勇于承认自己的无知，善于向别人求助，否则到头来，损失的时间和机会都是自己买单，得不偿失。**

2. 抓大放小，适当地忽略细节

在协作的过程中，团队总会在一些细节上产生冲突。如果任由团队成员在细节上争论不休，久而久之，就会影响团队之间的信任感。这个时候，就需要引导团队将注意力集中在大的方向上，适当地暂缓细节讨论，以保证团队的协作效率。

比如，一个业务逻辑是放在前端处理，还是放在后端处理，结果并没有太大区别，说白了，就是放在哪儿都行。但是，前端同学会坚持认为，逻辑处理都应该由后端来解决，以降低前端和业务的耦合性，这样说也没有错。可是，后端同学也会有自己的想法，比如针对前端拦截器的处理机制，后端到底要不要配合着返回前端要求的返回码，而不是直接抛出 http 原始的返回码呢？

类似的这些问题，没有谁对谁错之分，但是真要是纠结起来，也不是一两句话就能说清楚的。

这个时候，就需要有人拍板，选择一条更加符合常规的方式推进，并预留出后续的讨论空间。甚至，为了促进多地合作，自己人这边要适当地牺牲一些，以此来换取合作的顺利推进。这样一来，你会发现，有些不可调和的事情，在项目不断成功、人员不断磨合的过程中，也就不是个事情了。

总结

在这一讲中，关于如何开发产品，可以说，我是把自己在过去几个项目经历中的总结倾囊相授了。

其实，就像我在讲“DevOps 工程师需要的技能”中提到的那样，软实力（比如沟通协作、同理心、持续改进等）对促进产品快速迭代开发演进有着重大的作用。作为非专业产品经理，我也在慢慢地积累自己的产品心经，有机会再给你好好聊聊。

你可能还在想，最终千人的目标是否实现了呢？我想说的是，有些时候，真实生活比故事还要精彩。

就在预订目标的倒数第二天，平台用户只有 997 个。当时，我跟同事吐槽这个数字，他们说要不要拉几个用户进来，我说：“算了吧，随它去吧。”结果你猜怎样？在当天周五下班的时候，我又去平台上看了一眼，不多不少刚好 1000 个注册用户。当时我的第一感觉就是，**要相信，当我们把自己的全身心和热情都灌注在一个产品的开发过程中时，美好的事情会自然而然地发生。**

思考题

你对这一讲的哪部分内容印象最深刻呢？你有什么其他有助于产品快速研发落地的观点吗？

欢迎在留言区写下你的思考和答案，我们一起讨论，共同学习进步。如果你觉得这篇文章对你有所帮助，也欢迎你把文章分享给你的朋友。

© 版权归极客邦科技所有，未经许可不得传播售卖。 页面已增加防盗追踪，如有侵权极客邦将依法追究其法律责任。

上一篇　25 | 让数据说话：如何建设企业级数据度量平台？

下一篇　期中总结：3个典型问题答疑及如何高效学习

精选留言 (10)

写留言

心在飞
2019-12-18

印象最深：把自己的全部精力和热情投入到一件事情上的时候，结果通常不会太坏。
觉得石老师做的比较好的：产品需求、技术风险、开发流程、开发工具、团队建设。
产品需求：一开始有个初步的需求，不断迭代、清晰。
技术风险：能在项目早期消除该项目用到所有技术的技术风险

开发流程：敏捷软件开发，精益看板，明确分支策略、异地团队简间的沟通、合作机制...

展开

2

leslie

2019-12-17

内容很精辟：老师如果长期夹在产品与研发、销售与研发、研发与运维之中，整个的过程会更加的顺利的。电商的数年一直处于此种角色，由于工作中的随和且都做任何事情都绕不开需要数据部门做支持，导致长期在几方之间协调，不知不觉锻炼出了整体的格局观和效率沟通能力。

最近学习产品和项目管理课程时：自己就明显感受到这点。虽然之前没有专门的去学...

展开

1

陈斯佳

2019-12-18

“关于产品运营策略，‘酒香不怕巷子深’的理念已经有些过时了。想要一个产品获得成功，团队不仅要做得好，还要善于运营和宣传，而这又是技术团队的一大软肋。” 很赞成。对这个观点最有感触的时候是在我学习Linux的时候发现居然有这么那么有用的功能居然没人知道！

展开

陈斯佳

2019-12-18

“研发环境容器化”这真是一个很好的思路。真像老师所说的，在做环境部署的时候，文档再怎么详尽，都会被一两个不起眼的小坑绊倒，而且可能要很久才能再爬起来。现在想想敏捷宣言里面提到的“working software over comprehensive documentation”，放在运维文档也是一样，繁琐不变的文档永远都赶不上瞬息万变的环境...

展开

陈斯佳

2019-12-18

“事实证明，但凡能打硬仗的同事，在后来都是非常靠谱且独当一面的，这与年龄无关，哪怕是应届生，也同样如此。” 活的久了越来越发现，成长和年龄其实没有必然的联系。有些人虽然很年轻，但是有自己的思考框架和处事原则，积极主动愿意承担责任，事后会复盘，持续在迭代，这时候时间就是Ta的朋友；而有些人行事随性，做事挑肥拣瘦，逃避责任，得过且过，永远也不去反思，吃一堑，只长肉不涨智，这时候时间就是Ta的敌人。

展开

陈斯佳
2019-12-18

老师这篇文章真是看的让人热血沸腾！

展开

sugar
2019-12-18

不仅仅是学习devops的知识，更是一种思路和解决问题的方式方法

许童童
2019-12-17

很喜欢这种从无到有做一个产品的感觉，没有什么历史束缚，可以自由发挥自己的思想和主动性。

AlphaLiu
2019-12-17

看完这一篇，更加期待老师的实战篇！

展开

t86
2019-12-17

，期待下一讲石老师的实战历程

展开